RESEARCH ARTICLE OPEN

Automated Software Testing Using AI Tools

Kuldipsinh Jethwa [1], Mr Janak Maru[2]

1. (B Tech in Computer Engineering, Atmiya University, Rajkot, India
Email:jethwakuldipsinh293@gmail.com)

2. (Faculty of Engineering and Technology (CE), Atmiya University, Rajkot, India
Email: janak.maru@atmiyauni.ac.in)

Abstract- Software testing is a crucial step in software development. It ensures that applications function correctly and meet user requirements. Traditional automated testing tools like Selenium and JUnit are popular but often encounter issues such as fragile test cases and high maintenance costs when applications change frequently. Recently, Artificial Intelligence (AI) has been introduced into software testing to address these challenges. AI-powered testing tools can automatically generate test cases, adjust to changes in the application, and even identify visual differences in user interfaces. This research paper examines how AI tools improve automated software testing compared to traditional methods. A case study approach compares the efficiency, accuracy, and maintenance effort of AI-based testing tools with traditional ones. The study indicates that AI testing can decrease manual effort and enhance flexibility, but it also faces limitations, including higher costs and fewer free tools available. Overall, AI-powered testing has great potential to make software testing smarter, faster, and more reliable in the future.

1. Introduction

Software development has become increasingly complex. Applications are growing larger and require frequent updates. To maintain quality, software testing is essential. Manual testing takes time and is prone to human error. That's why automated testing tools like Selenium, JUnit, and Cypress are popular. These tools allow developers to write test scripts that can be reused and run automatically. However, traditional automation often struggles when applications frequently change. Test cases may fail due to minor updates in the interface, leading to increased maintenance efforts.

Artificial Intelligence (AI) offers new opportunities to improve software testing. AI-powered tools can learn from previous tests, adjust to changes, and automatically create new test cases. This paper explores the role of AI in automated software testing, contrasting it with traditional tools and examining its benefits, challenges, and future possibilities.

2. Background and Literature Review

2.1 Importance of Software Testing

- Ensures software quality.

- Detects errors before release.

- Increases customer satisfaction.

- Saves costs and time in the long term.

2.2 Traditional Automated Testing

Traditional automated testing involves writing scripts that run test cases repeatedly without manual intervention. Tools like Selenium for web applications, JUnit for Java programs, and Cypress for modern front-end frameworks are widely used. These tools provide speed and reusability but face challenges in scalability, maintenance, and adaptability.

2.3 Limitations of Traditional Testing

- Fragile test scripts that break when the user interface changes.

- High maintenance costs for updating test scripts.

- Limited ability to handle visual testing or dynamic UI changes.

- Dependency on human testers for test design.

2.4 AI in Software Testing

AI techniques like machine learning, natural language processing, and computer vision are being integrated into testing. AI-based tools like Testim, Applitools, and Mabl can:

- Automatically generate test cases.

- Perform visual testing with image recognition.

- Adapt to minor UI changes through self-healing tests.

- Predict high-risk areas in the code for targeted testing.

2.5 Related Work

Several studies highlight AI's growing role in testing. For instance:

- Applitools uses computer vision to detect visual bugs that traditional tools overlook.

- Testim employs AI to reduce flakiness in test cases.

- Mabl integrates AI to enhance cloud-based intelligent test automation.

Researchers argue that while AI can improve reliability, it still lacks the maturity needed for very large projects.

3. Research Objectives

The objectives of this study are:

1. To analyze differences between traditional automated testing and AI-based testing tools.

2. To evaluate the efficiency, accuracy, and adaptability of AI tools.

3. To identify the limitations and challenges of using AI in software testing.

4. To offer recommendations for students and small teams considering AI-based testing.

–

4. Methodology

This research employs a comparative case study method:

- Case Selection: A small web application with login and form submission features was chosen.

- Tools Used:

  - Traditional: Selenium, JUnit.

  - AI-based: Testim, Applitools (trial versions).

- Evaluation Metrics:

  - Test execution time.

  - Number of defects detected.

  - Effort needed for maintenance.

  - Adaptability to UI changes.

- Data Collection: Experiments are conducted over three weeks, recording failures, test execution logs, and time spent on test maintenance.

---

5. Case Study and Results

5.1 Traditional Tools

Using Selenium and JUnit, test cases were made for login validation and form submission. Initially, the tools worked well, but they failed when minor changes were made to the UI (e.g., changes in button names). Fixing the scripts took extra time, raising maintenance efforts.

5.2 AI-Based Tools

With Testim and Applitools, tests automatically adjusted to minor UI changes. Applitools identified visual bugs like misaligned buttons, which Selenium could not catch. The AI tools needed less maintenance and produced more accurate results in visual testing.

5.3 Comparative Results Table

Metric	**Traditional Tools (Selenium/JUnit)**	**AI-Based Tools (Testim/Applitools)**
Execution Time	10 min	10 min
Bugs Detected	8	12
Maintenance Effort	High	Low
Adaptability to UI Changes	Poor	High
Ease of Use	Medium	Easy
Cost	Free/Open-source	Paid/Trial Available

5.4 Summary Table of Key Findings

Category	Traditional Tools	AI-Based Tools
Reliability	Good, but scripts break often	Very Good, adapts automatically
Visual Testing	Limited	Strong visual bug detection
Learning Curve	Easy for beginners	Moderate (AI concepts needed)
Best Use Case	Stable applications	Applications with frequent changes

6. Discussion

The results indicate that AI-based tools offer major benefits in adaptability and bug detection. For small projects, AI testing lightens the load of maintaining test scripts. However, AI tools typically come with subscription fees and limited free access, which can pose a challenge for students or small organizations.

Advantages of AI Testing

- Reduces manual effort.

- Automatically manages UI changes.

- Improves bug detection, particularly for visual issues.

- Saves maintenance time.

Disadvantages of AI Testing

- High cost of commercial tools.

- Limited free or open-source options.

- Some false positives may occur.

- Learning curve for testers.

Traditional tools still matter because they are open-source, widely supported, and effective for stable applications. A combined approach that utilizes traditional tools for functional testing and AI tools for adaptability can yield the best results.

---

7. Challenges and Limitations

- Cost: Many AI-based testing tools are commercial.

- Learning Curve: Testers need to grasp both testing and AI concepts.

- Limited Open-Source Options: Most AI-powered tools are proprietary.

- Scalability Issues: AI tools may still struggle with very large and complex systems.

_

8. Future Scope

AI in software testing is still progressing. Future research could focus on:

- Development of open-source AI testing frameworks.

- Better integration of AI testing with DevOps and CI/CD pipelines.

- Smarter self-healing algorithms to reduce false positives.

- Combining AI with human testers for maximum effectiveness.

_____

_

9. Conclusion

This research indicates that AI-powered automated testing tools provide clear advantages over traditional methods in adaptability, maintenance, and bug detection. While traditional tools are still valuable for their cost-effectiveness and reliability, AI tools introduce innovation and efficiency into the testing process. Despite their limitations, AI tools present a promising direction for the future of software testing, making it smarter, faster, and more reliable.

_____

_

References

1. SeleniumHQ. (2025). Selenium Documentation. https://www.selenium.dev/

2. JUnit Team. (2025). JUnit User Guide. https://junit.org/junit5/

3. Testim.io. (2025). AI-based Test Automation. https://www.testim.io/

4. Applitools. (2025). Visual AI for Automated Testing. https://applitools.com/

5. Mabl. (2025). Intelligent Test Automation. https://www.mabl.com/

6. Research papers on AI in Software Testing (2020–2024).

